PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/60 — A2

(11) International Publication Number: WO 00/65507

(43) International Publication Date: 2 November 2000 (02.11.00)

(21) International Application Number: PCT/US00/10782

(22) International Filing Date: 24 April 2000 (24.04.00)

(30) Priority Data:
60/130,568 22 April 1999 (22.04.99) US

(71) Applicant: NETWORK SOLUTIONS, INC. [US/US]; 505 Huntmar Park Drive, Herndon, VA 20170 (US).

(72) Inventor: SRIVASTAVA, Manoj; 100 Rock Haven Road, E-306, Carrboro, NC 27510 (US).

(74) Agents: GARRETT, Arthur, S. et al.; Finnegan, Henderson, Farrabow, Garrett & Dunner, LLP, 1300 I Street, N.W., Washington, DC 20005-3315 (US).

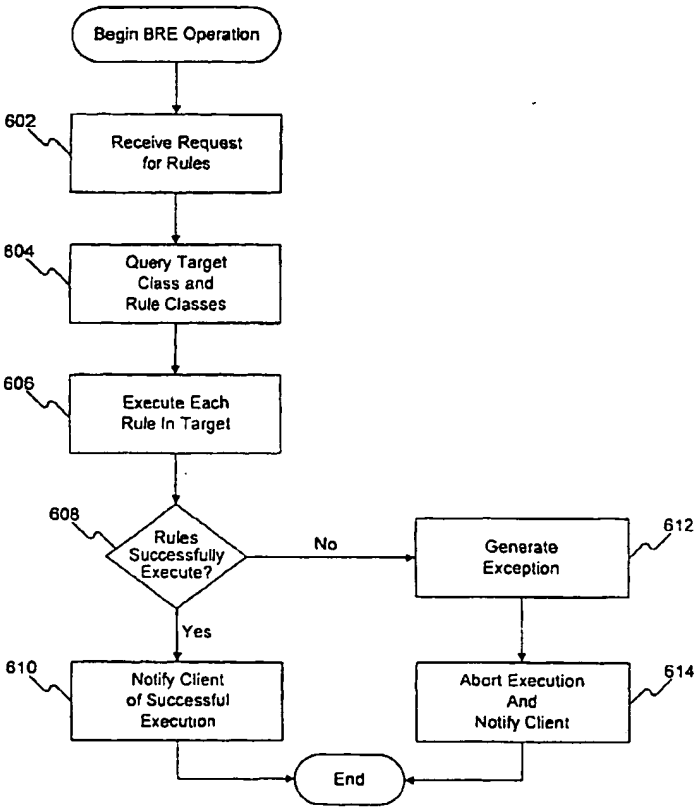
(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: BUSINESS RULE ENGINE

(57) Abstract

Methods and systems consistent with the present invention solve the inherent problems with existing systems that incorporate business rules by providing a business rule engine (BRE) that serves as a business rule repository for business rules. Specifically, the BRE maintains a collection of both rules and targets and an association between the rules and targets for e-commerce applications. Each time a client accesses an e-commerce application which triggers a business procedure (target) to execute a set of policies (rules), the application queries the BRE for a list of rules, and an order to apply the rules in response to the client's request. The BRE also provides a facility to modify, add, or delete both business rules and targets and modify the order of execution by the application.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

BUSINESS RULE ENGINE

RELATED APPLICATIONS

Provisional U.S. Patent Application No. 60/130,568 entitled "Shared Domain Name Registration System," filed April 22, 1999, is relied upon and is incorporated by reference in its entirety in this application.

BACKGROUND OF THE INVENTION

A. Field of the Invention

This invention relates generally to data processing systems and, more particularly, to business rules.

B. Description of the Related Art

With the explosive growth of the Internet, businesses are attempting to modernize their business services. Using the Internet, businesses can provide their business services to customers that previously were unreachable. One way business now can reach their customers is with computer software that provides their business services "online," such as a business application. Examples of business applications may be e-commerce applications, such as an online bookstore, training site, or a domain name registry site, or an accounting application.

Providing business services to customers in the "cyberworld" is no different from that of providing services to customers in the real world. Just as businesses adhere to policies and procedures when dealing with real world customers (e.g., requiring a customer to present two forms of identification), businesses also follow the same policies and procedures when dealing with "online" customers. However, instead of human interaction, the business application uses business logic, in the form of software logic, to designate online policies and procedures (e.g., requiring a customer to enter a valid 10 digit identification number). Multiple policies in a particular order in business applications are known as business procedures. For example, in the case of domain name registry services, a business policy may be a verification of a customer identification, or a query for a new nameserver, and the business procedure may be an ordering of a new domain name.

Existing business applications contain business logic to create business policies and procedures, however business policies and procedures tend to change over a period of time to accommodate business modifications. Business policies may change over time depending upon various market conditions, and/or business decisions by the business's management. For example, in the case of a domain name registry services, to register a domain name with a registry, such as Network Solutions, the registrar must pay a small yearly fee (e.g., $35). However, the yearly fee may fluctuate, or even be waived in certain circumstances. In this case, the pricing of the domain name may be a business policy, and the registering of the domain name may be a business procedure.

Therefore, although business software developers can incorporate business policies and procedures into business applications, and even incorporate modifications to the policies and procedures, the developers are limited in that they must hard-code the policies and procedures as logic into the business applications. Thus, when business policies change and they need to be modified, added, or deleted in the business applications, the developer must first locate each affected business policy and the procedure from where the rule is invoked. Only then may the developer modify the policy to incorporate the modifications. Moreover, if the business policies are not discrete (e.g., several business policies are intertwined in their implementation), it could be very difficult to change the existing business policy, even possibly requiring a rewrite of all business logic.

There is therefore a need for a system that can facilitate manipulation of business policies within an application. Such a system not only allows adding, modifying, or deleting business policies with little or no modification in an application's source code, but also it easily defines what business policies are implemented by the application and allows execution of business policies from within the application's source code.

SUMMARY OF THE INVENTION

Methods and systems consistent with the present invention solve the inherent problems with existing systems that incorporate business rules by providing a business rule engine (BRE) that serves as a business rule repository for business rules. Specifically, the BRE maintains a collection of both rules and targets and an association between the rules and

targets for e-commerce applications. Each time a client accesses an e-commerce application which triggers a business procedure (target) to execute a set of policies (rules), the application queries the BRE for a list of rules, and an order to apply the rules in response to the client's request. The BRE also provides a facility to modify, add, or delete both business rules and targets and modify the order of execution by the application.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an implementation of the invention and, together with the description, serve to explain the advantages and principles of the invention. In the drawings.

Figure 1 depicts a data processing system suitable for practicing methods and systems consistent with the principles of the present invention;

Figure 2 depicts a more detailed diagram of the client depicted in Fig. 1;

Figure 3A depicts a more detailed diagram of the application server depicted in Fig. 1; Figure 3B depicts a more detailed diagram of the business rule server depicted in Fig. 1;

Figure 4 depicts an exemplary database for use with methods and systems consistent with the principles of the present invention;

Figure 5 depicts a flow chart of the steps performed by the business rule server of Fig. 1 when initializing the business rule engine; and

Figure 6 depicts a flow chart of the steps performed by the business rule server of Fig. 1 when providing business rules to applications in a manner consistent with the present invention.

DETAILED DESCRIPTION

The following detailed description of the invention refers to the accompanying drawings. Although the description includes exemplary implementations, other implementations are possible. and changes may be made to the implementations described without departing from the spirit and scope of the invention. The following detailed description does not limit the invention. Instead, the scope of the invention is defined by the

appended claims. Wherever possible, the same reference numbers will be used throughout the drawings and the following description to refer to the same or like parts.

Overview

Methods and systems consistent with the present invention provide a Business Rule Engine (BRE) that enables applications to easily add, delete, or modify business logic without having to modify existing application code. The BRE maintains rules and targets externally from that of the application, instead of hard-coding the rules within the application's source code. A rule is a business policy and multiple rules in a particular order is a target. A business application, such as an e-commerce application or an accounting application, may request business rules and targets from the BRE to execute business procedures and policies. The BRE may contain an interface BRE class to handle requests for rules and targets from the applications.

The BRE contains both targets and rules, and associations between the targets and rules. Now will be explained the class structure of rules, targets and the BRE. Although a specific class structure for targets, rules, and the BRE are defined, one skilled in the art will appreciate that other methods, objects, and/or class structures may exist for rules, targets and the BRE classes.

The BRE is a concrete singleton class that may implement various operations. A singleton class is a type of class that cannot be duplicated. Since singletons are instantiated only once, instead of each method instantiating its own copy of the BRE, a reference (handle) to the BRE is passed from one method to another method.

The BRE may contain methods such as init, getRules, getAllRules, and getAllTargets. The init method initializes the BRE. As part of initialization the init method reads from a secondary storage device all targets, rules and associations between targets and rules. The getRules method returns all rules associated with a target that have not expired to an application. Rules may contain an expiration date and once expired, the rule may no longer be used. For example, a rule may apply to a business promotional period for two months (e.g., a discounted domain registration fee). The get AllRules method returns all

rules, regardless of their expiration status to an application. Finally, the getAllTarget method returns all targets that are defined for the application.

A rule is an implementation of one business policy and may be identified by a descriptive name. For example, a rule may be to check if the country code is valid, or if a user has authorization to perform a particular operation. Each rule defines a specific decision to be taken by the application. A rule can be explicitly associated with a target or it can be associated with multiple targets. There are multiple types of rules, such as boolean, modifier, and database. A boolean rule checks for a condition. If the condition evaluated is "true," then the rule has passed. Conversely, if the condition is "false," then the rule has failed. A modifier rule computes a value and assigns that value to a variable or an class. The database rule removes, retrieves, or modifies an object, a class, or a record from some secondary storage.

A rule is an abstract class from which all other rules will inherit. A rule class may contain various methods that will be inherited by the inheriting class, such as getType, getDescription, execute, and init. The getType method returns the type of rule (e.g., boolean, modify, or database). The getDescription method returns a detailed description of the business policy implemented by the rule (e.g., check email status). The execute method executes the rule in the application. The init method initializes the rule before it is invoked. Rules are initialized once at the time of instantiation. A target is an identifier that identifies an ordered collection of rules to be invoked by a business application as a business procedure and may be referenced by a descriptive name. Targets are mapped to business procedures for which multiple rules have been defined. For example, in the case of a registry application, a target may be a domain name transfer from one entity to another entity. There may be several rules associated with a target. A target identifies all of the rules in a specific order associated with that target.

A target is a concrete class that is instantiated in the BRE as a private class. The target class may contain methods, such as init and getRules. The init method initializes the target class with all associated rules. The getRules method returns an ordered collection of rule classes for a given target. Each time the BRE instantiated, all target classes are

instantiated as well. Each target class that is instantiated contains a list of rules IDs. As explained, the rules are inherited from a base rule and are instantiated as a singleton class.

System Components

Fig. 1 depicts a data processing system 100 suitable for practicing methods and systems consistent with the present invention. Data processing system 100 comprises a client 102, application server 104, and a business rule server 106, all connected together in a network 110, such as the Internet. A user uses client 102 to request information from application server 104. In the case of a domain name registry business, a user may request business targets, such as registration of a domain name, or change of ownership of a domain name.

Application server 104 may be a business application server that interfaces with clients 102 using Web documents. Application server 104 may also interface with business rule server 106 using any communication programming interface such as a object oriented programming interface. Application server 104 and business rule server 106 may be located at a businesses processing facility.

Figure 2 depicts a more detailed diagram of client 102, which contains a memory 210, a secondary storage device 220, a central processing unit (CPU) 230, an input device 240, and a video display 250. Memory 210 includes browser 212 that allows users to interact with application server 104 by transmitting and receiving files, such as Web documents. An example of browsers suitable for use with methods and systems consistent with the present invention are the NETSCAPE NAVIGATOR browser, from Netscape Corp, or the INTERNET EXPLORER browser, from Microsoft Corp.

As shown in Figure 3A, application server 104 includes a memory 320, a secondary storage device 324, a CPU 326, an input device 328, and a video display 330. Memory 320 includes a business application 322 that provides an application, such as an e-commerce, Web. or other business application to users. For example, business application 322 may be an online store, a domain name registry service, or any type of application that contains business logic. Business application 322 may also contains well-known Web server software (not shown) to transmit and receive files (e.g., Web documents) to the user.

As shown in Figure 3B, business rule server 106 includes a BRE 352, a secondary storage device 360, a CPU 364, an input device 366, and a video display 368. BRE 352 (stored in a memory not shown) maintains a dynamic listing of all rules and targets, and associations between the rules and targets. BRE 352 also responds to queries from application 322 through any well-known interface (not shown), such as Java servlets (or other comparable object oriented programming interface), Web interface or Application Program Interface (API). A servlet is a Java applet that runs on a server. An API is a set of routines, protocols, or tools for communicating with software applications. APIs provide efficient access to BRE 352 without the need for additional software to interface with the engine. BRE 352 contains rule classes 354, target classes 356, and a global BRE class 358, all described below.

Secondary storage device 360 contains a database 362. Database 362 contains a listing of all rules and targets, and an association (mapping) of ordered listings of rules and targets. BRE 352 loads targets and rules and their associations from database 362 each time BRE 352 is initialized. In addition, developers may access database 362 to add, modify, or change rules, targets, or their associations. To access database 362, a developer may use any well-know database interface (not shown), such as the MICROSOFT ACCESS database, from Microsoft Corp., or the ORACLE database from Oracle Corp. Each time a developer modifies database 362, BRE 352 may be reinitialized using the initialization procedure (described below).

Database 362 may be a relational database, with three tables including, a target table, a rule table, and an association table. An exemplary representation of database 362 is depicted as three tables in Figure 4.

As shown in Figure 4, database 362 contains a target table 400, a rule table 410, and an association table 420. Target table 400 contains all available targets for BRE 352. Each record 402 in target table 400 contains a target name and target ID. The target name is the name of the target (e.g.,"Verify_User") and the target ID is a unique target identifier (e.g., "A"). Rule table 410 contains executable rules available for application 322. Each record 412 contains a class name, rule ID and rule type. Class name is the rule class type (e.g.,

"Check_Auth"). The rule ID is a unique rule identifier (e.g., "1"). The rule type is the type of rule (e.g., database, boolean, modifier). Finally, association table 420 contains a link between rule table 410 and target table 400. Each record 422 in association table 420 contains a target ID, rule ID, and a sequence number. The sequence number indicates the order in which a rule will be executed within the target. For example, record 422 indicates that rule "1" executes third in target "A."

In an alternative embodiment, database 362 may be a flat file database in a Comma Separated Values (CSV) format. The CSV format stores records as a flat file in a text format. Each field in CSV format is followed by a comma. Thus, a target may be a first entry, and the rules (in order of execution) may be in following entries followed by a comma.

Initializing the BRE

As shown in Figure 5, the initialization of BRE 352 and associated classes are initiated, for example, by first loading targets and rules from database 362 (step 502). The init method class may be used to load the rules and targets as rule classes 354 and target classes 356. The init method creates a pointer for each rule and each target that is loaded from database 356. To create the pointers for both rule classes 354 and target classes 356, the BRE init method invokes an init method for each rule class 354 and target class 356. Classes for both rules and targets may be maintained in BRE 352.

Next, each rule class 354 may be initialized (step 504). For each rule class 354 the init method initiates in step 502, a singleton rule object is instantiated for that rule class 354. That is, using pointers associated with rule classes 354, each rule class 354 loads accompanying data (e.g., class name, rule ID, and rule type) for each class 354 from database 362. With the accompanying data from database 362, each rule class 354 may instantiate a new singleton rule object according to the class name. For example, if the class name is "Transfer_Domain_Rule," a singleton rule object of this type is instantiated. Instead of accessing each rule class 354 directly, BRE 352 and/or target classes 356 may access the singleton rule object version of the rule class.

Once the singleton rule objects are all instantiated, each target class 356 may be initialized (step 506). To initialize each target class 356, BRE 352 associates an ordered

collection of rules with a target class 356. BRE 352 obtains the order of the rules from association table 420 in database 362. Instead of loading the actual rule classes 354, target class 356 loads a pointer associated with each rule class 354 and a sequence number (obtained from database 362).

In the last step of initialization, BRE 352 may initialize a global BRE class 358 (step 508). In this step, BRE class 358 is instantiated as a singleton global class. A global class is a type of class known to all applications at run-time. In addition, BRE class 358 contains a handle that provides access to BRE 352 from applications 322. Once BRE 352 and classes 354-358 are initialized, the handle associated with BRE class 358 may be transmitted to application 322. (step 510) Thus, each time application 322 requires information from BRE 352, a calling method from application 322 may use the handle to locate BRE class 358. For example, a method located in an application object may request information relating to available targets using the handle and the getAllTargets method.

Business Rule Engine Operation

Figure 6 depicts a flow chart of the steps performed each time an application 322 requests information from BRE 352, such as an ordered list of rules to apply to a known target. In the first step, BRE 352 receives a request for an ordered set of rules from application 322 (step 602). The request may include a BRE handle and a target descriptor, such as a target ID. Application 322 may obtain the BRE handle as a returned parameter from BRE class 358 after initializing BRE 352, or by communicating with a local method that has previously initialized BRE 352.

Next, BRE 352, using BRE class 358, queries a target class 356 that corresponds to the target requested by application 322 (step 604). BRE 352 queries a target class 356 for a listing of all rules associated with that target class 356. In response to the query, target class 356 may return an ordered listing of rule IDs. BRE 352 may use the rule IDs to query various singleton rule objects for detailed information (e.g., rule class, rule type). BRE 352 then returns the information received from target class 366 and the singleton rule objects to application 322. In addition, application 322 may request a listing of all available rules (getAllRules method), nonexpired rules (getRules method), or all available targets

(getAllTargets method). One skilled in the art will appreciate that application 322 may query BRE 352 for other information regarding rules, targets, and associations between the rules and targets.

As each rule class is transmitted to application 322, application 322 executes the rule (step 606). If all rules execute successfully (step 608), a notification is sent to the client informing the client of the successful execution of the target and associated rules (step 610). If however, the rules do not execute successfully (step 608), a rule exception is generated (step 612), and an abort command is sent to the client (step 614). Thus, each rule is executed without having to modify any source code related to application 322.

Conclusion

As described, methods and systems consistent with the present invention provide a business rule engine (BRE) that serves as a business rule repository for business rules.

Although aspects of the present invention are described as being stored in memory, one skilled in the art will appreciate that these aspects may be stored on or read from other computer readable media, such as secondary storage devices, like hard disks, floppy disks, and CD-ROM; a carrier wave received from a network like the Internet; or other forms of ROM or RAM. Additionally, although specific components and programs of client computer 102, and various servers have been described, one skilled in the art will appreciate that these may contain additional or different components or programs.

The foregoing description of an implementation of the invention has been presented for purposes of illustration and description. It is not exhaustive and does not limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practicing of the invention. For example, the described implementation includes software but the present invention may be implemented as a combination of hardware and software or in hardware alone.

WHAT IS CLAIMED IS:

1. A method for providing business rules to applications, comprising the steps, executed in a data processing system, of:

maintaining a business rule engine, wherein the engine contains information reflecting rules and targets, and associations between the rules and targets; and

permitting applications to obtain an ordered set of rules through the business rule engine.

2. The method of claim 1, wherein each target corresponds to a business procedure, and wherein each rule corresponds to a business policy.

3. The method of claim 1, wherein maintaining a set of rules, further comprises the step of maintaining the rules in a location separate from that of application code associated with the applications.

4. The method of claim 1, further comprising the step of providing an interface, such that each application can use a set of methods to receive information corresponding to the rules and targets.

5. The method of claim 4, wherein permitting applications to obtain an ordered set of rules further comprises the steps of:

receiving a request for a list of rules from an application, the request including information associated with a target and a handle associated with the business rule engine; and

querying a target class for list of rule associated with the target based on the request.

6. A method for maintaining a business rule engine, comprising the steps, executed in a data processing system, of:

providing a set of rules to the business rule engine, wherein each rule corresponds to a business policy;

providing a set of targets to the business rule engine, wherein each target corresponds to a business procedure;

providing an association between a set of rules and each target to the business rule engine; and

wherein the business rule engine is separate from the application and provides the rules and targets, and the associations between a set of rules and each target to a plurality of applications.

7. The method of claim 6, wherein each rule is stored as a singleton object.

8. The method of claim 6, wherein the business rule engine contains a business rule engine class that accepts requests for targets and rules from applications, and wherein the business rule engine class provides a handle associated with the business rule engine to applications.

9. A computer implemented method for providing business rules and targets from a business rule engine to applications, comprising the steps, executed in a data processing system, of:

initializing the business rule engine;

receiving requests from an application for an ordered set of rules based on a target; and

providing the ordered set of rules to the requesting application, such that the application can execute the set of rules in a specified order.

10. The method of claim 9, wherein initializing the business rule engine further comprises the steps of:

loading targets and rules from a database;

initializing a rule class for each rule;

instantiating a singleton rule object for each rule class;

initializing a target class for each target, wherein each target class is associated with an ordered set of rules; and

initializing a global business rule class, wherein the business rule class provides a handle associated with the business rule engine to the application.

11. The method of claim 9, further comprising the steps of:

executing each rule; and

determining if all rules execute successfully, and if so, transmitting a notification to a client.

12. The method of claim 11, further comprising the step of generating an exception if all rules do not execute successfully.

13. The method of claim 9, wherein receiving requests from an application further comprises the steps of:

receiving a request including a target identification and a handle associated with the business rule engine; and

querying a target class associated with the target identification.

14. The method of claim 9, further comprising the steps of:

receiving requests from an application for a listing of all rules, expired rules, or all targets associated with the application; and

providing results of the request to the requesting application.

15. A system for maintaining a business rule engine, comprising:

providing means for providing a set of rules to the business rule engine, wherein each rule corresponds to a business policy;

providing means for providing a set of targets to the business rule engine, wherein each target corresponds to a business procedure;

providing means for providing an association between a set of rules an each target to the business rule engine; and

wherein the business rule engine is separate from the application and provides the rules and targets, and the associations between a set of rules and each target to a plurality of applications.

16. The system of claim 15, wherein each rule is stored in a memory as a singleton class.

17. The system of claim 15, wherein the business rule engine contains a business rule engine class that accepts requests for targets and rules from applications, and wherein the business rule engine class provides a handle associated with the business rule engine to applications.

18. A system for providing rules and targets from a business rule engine to applications, comprising:

initializing means for initializing the business rule engine;

receiving means for receiving requests from an application for an ordered set of rules based on a target; and

providing means for providing the ordered set of rules to the requesting application, such that the application can execute the set of rules in a specified order.

19. The system of claim 18, wherein the means for initializing further:

loads targets and rules from a database;

instantiates a singleton rule class for each rule;

initializes a target class for each target, wherein each target class is associated with an ordered set of rules; and

initializes a global business rule class, wherein the business rule class provides a handle associated with the business rule engine to the application.

20. The system of claim 18, further comprising:

executing means for executing each rule; and

determining means for determining if all rules execute successfully, and if so, transmitting a notification to a client.

21. The system of claim 20, further comprising generating means for generating an exception if all rules do not execute successfully.

22. The system of claim 18, wherein the receiving means further includes:

receiving means for receiving a request including a target identification and a handle associated with the business rule engine; and

querying means for querying a target class associated with the target identification.

23. The system of claim 18, further comprising:

receiving means for receiving requests from an application for a listing of all rules, expired rules, or all targets associated with the application; and

providing means for providing results of the request to the requesting application.

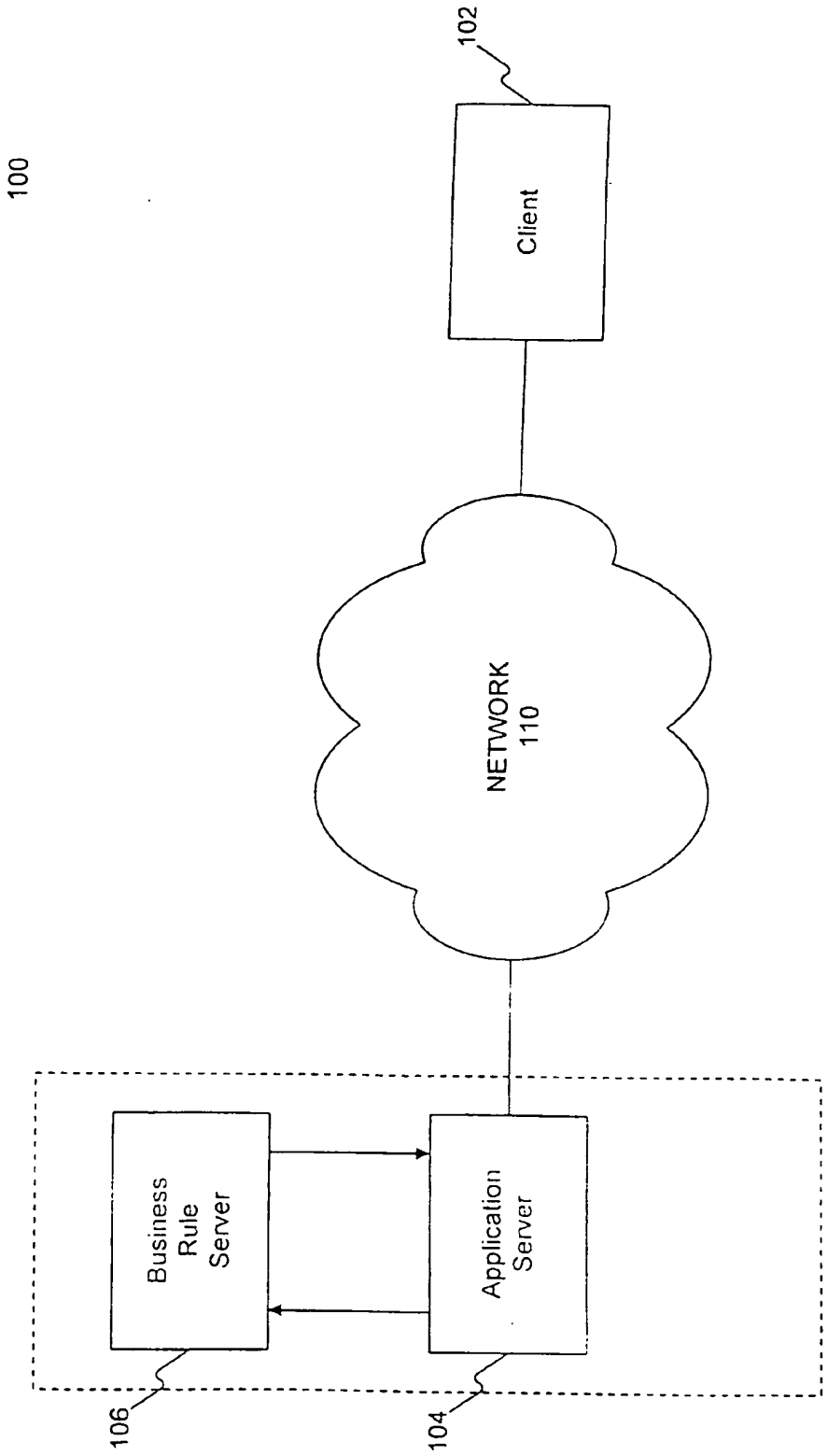

100
102
Client
NETWORK
110
Business
Rule
Server
Application
Server
106
104

FIG. 1

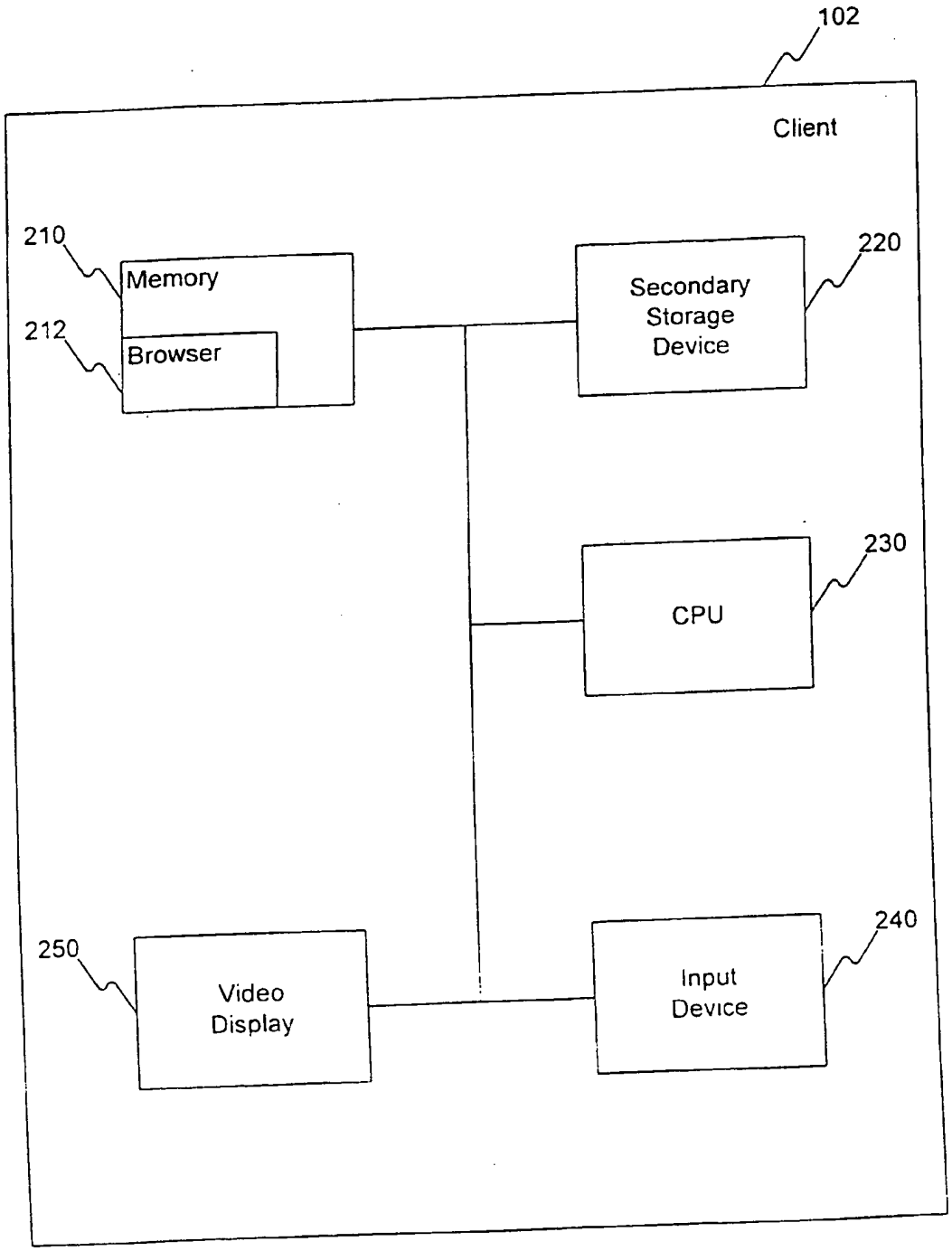

102
Client
210
Memory
212
Browser
220
Secondary Storage Device
230
CPU
240
Input Device
250
Video Display

FIG. 2

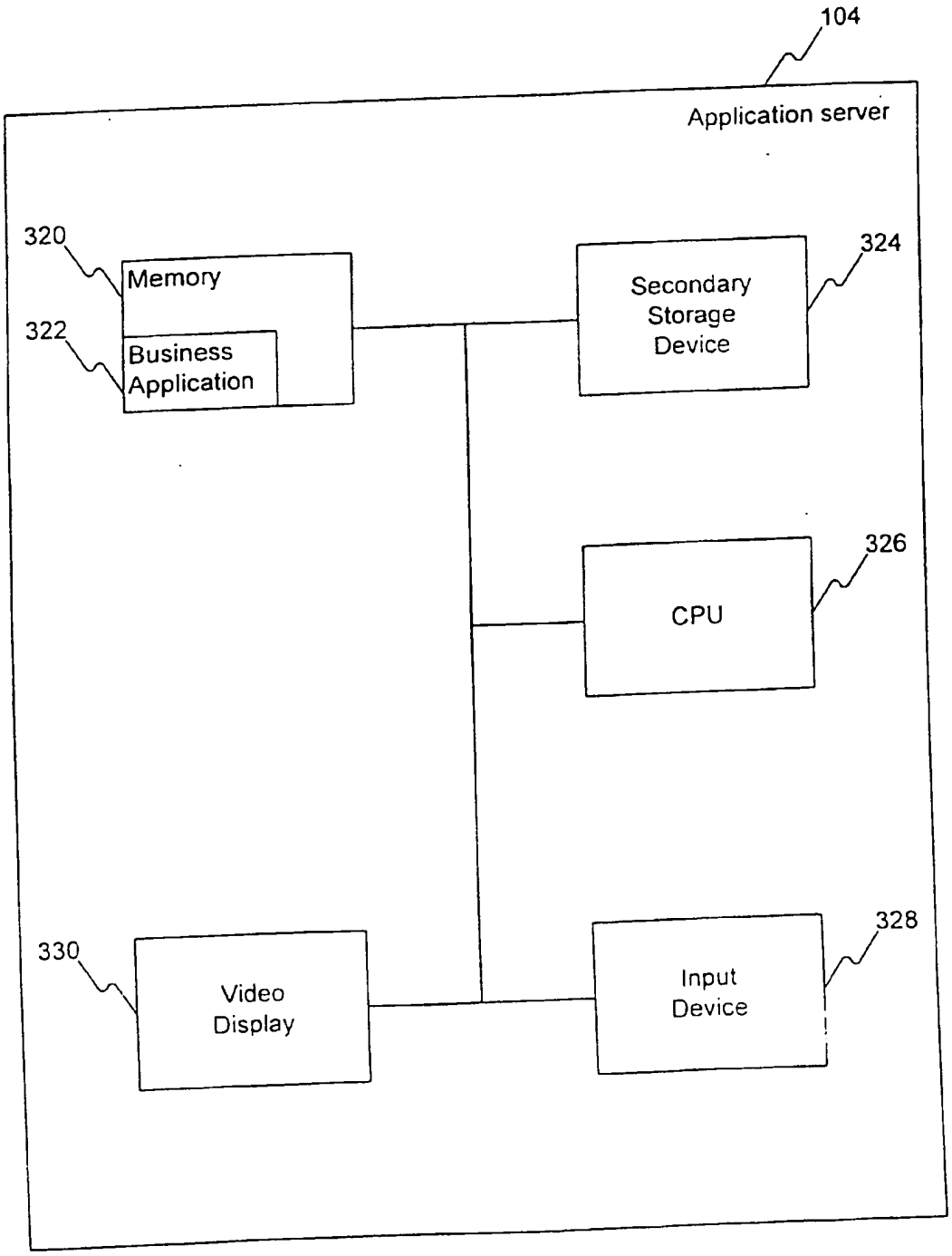

104
Application server
320
Memory
322
Business Application
324
Secondary Storage Device
326
CPU
328
Input Device
330
Video Display

FIG. 3A

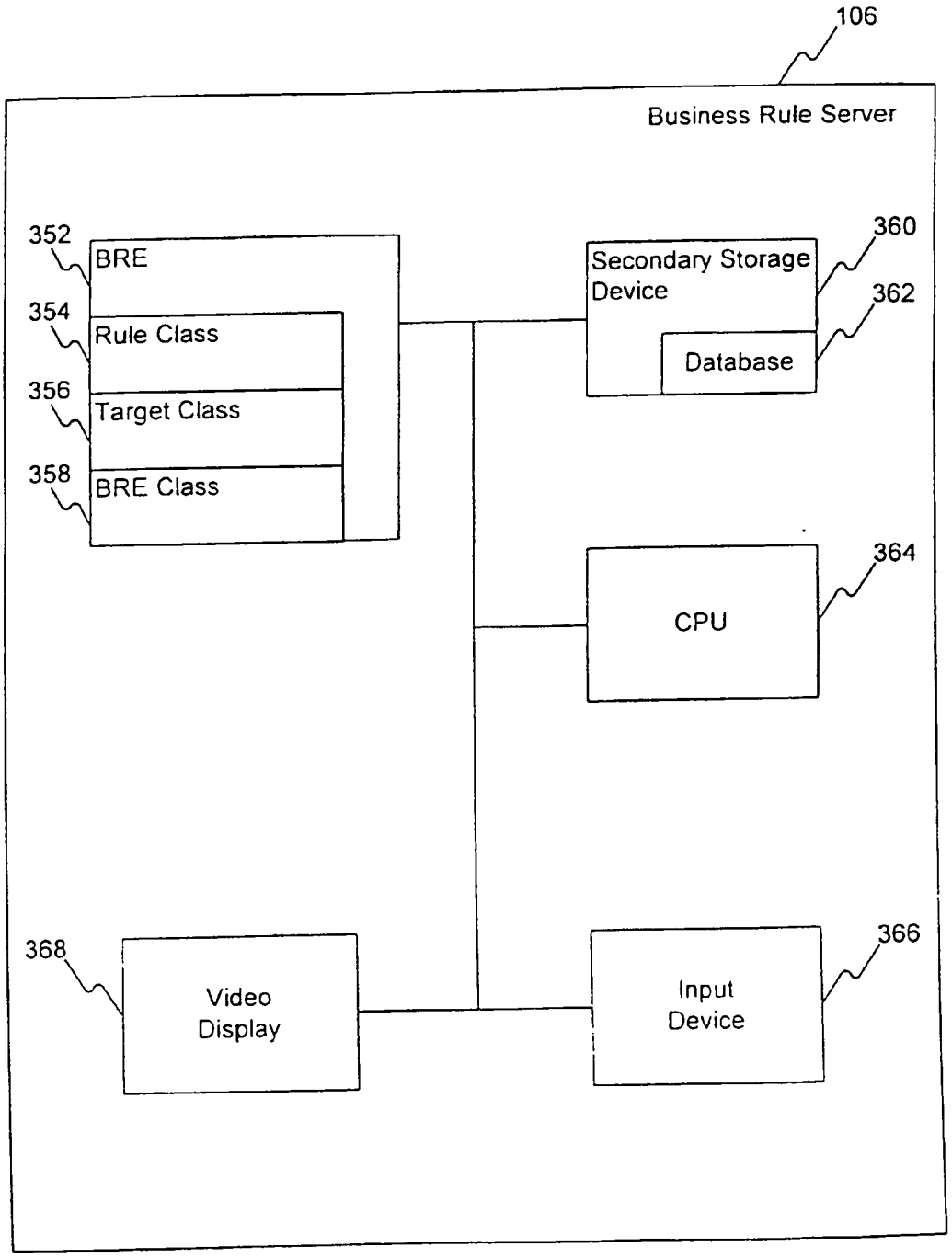

106
Business Rule Server
352
BRE
354
Rule Class
356
Target Class
358
BRE Class
360
Secondary Storage Device
362
Database
364
CPU
366
Input Device
368
Video Display

FIG. 3B

362

400

Target Table

Target Name	Target ID
Verify _User	A

402

410

Rule Table

Class Name	Rule ID	Rule Type
Check_Auth	1	Boolean
Order_Exists	2	Boolean
Chk_E-Mail	3	Boolean
Funds_Sufficient	4	Boolean
Permit_Order	5	Modifier

412

420

Association Table

Target ID	Rule ID	Seq #
A	1	3
A	2	1
A	3	2

422

FIG. 4

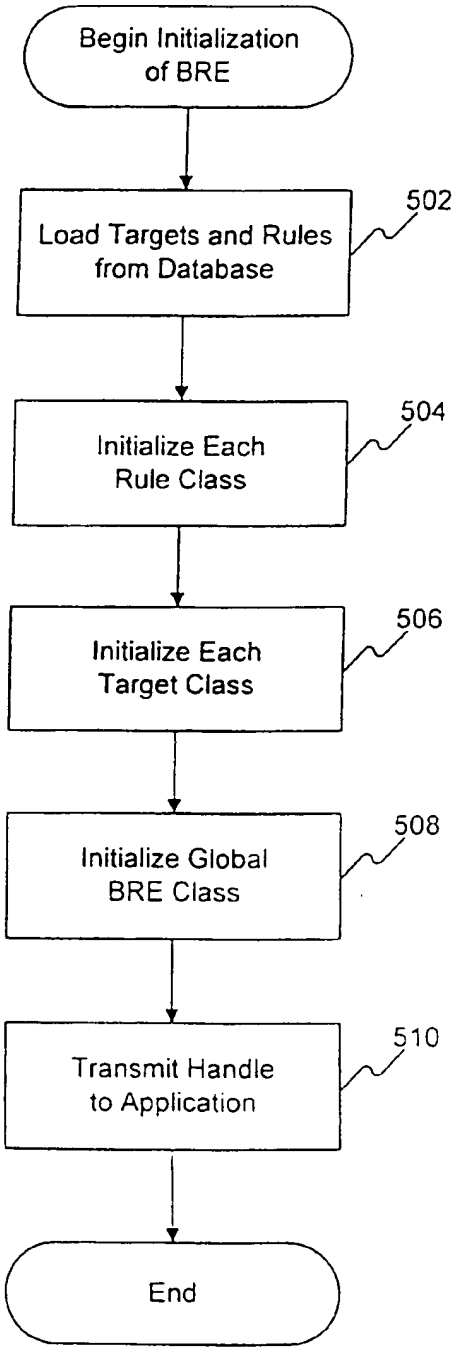

Begin Initialization of BRE
Load Targets and Rules from Database
502
Initialize Each Rule Class
504
Initialize Each Target Class
506
Initialize Global BRE Class
508
Transmit Handle to Application
510
End

FIG. 5

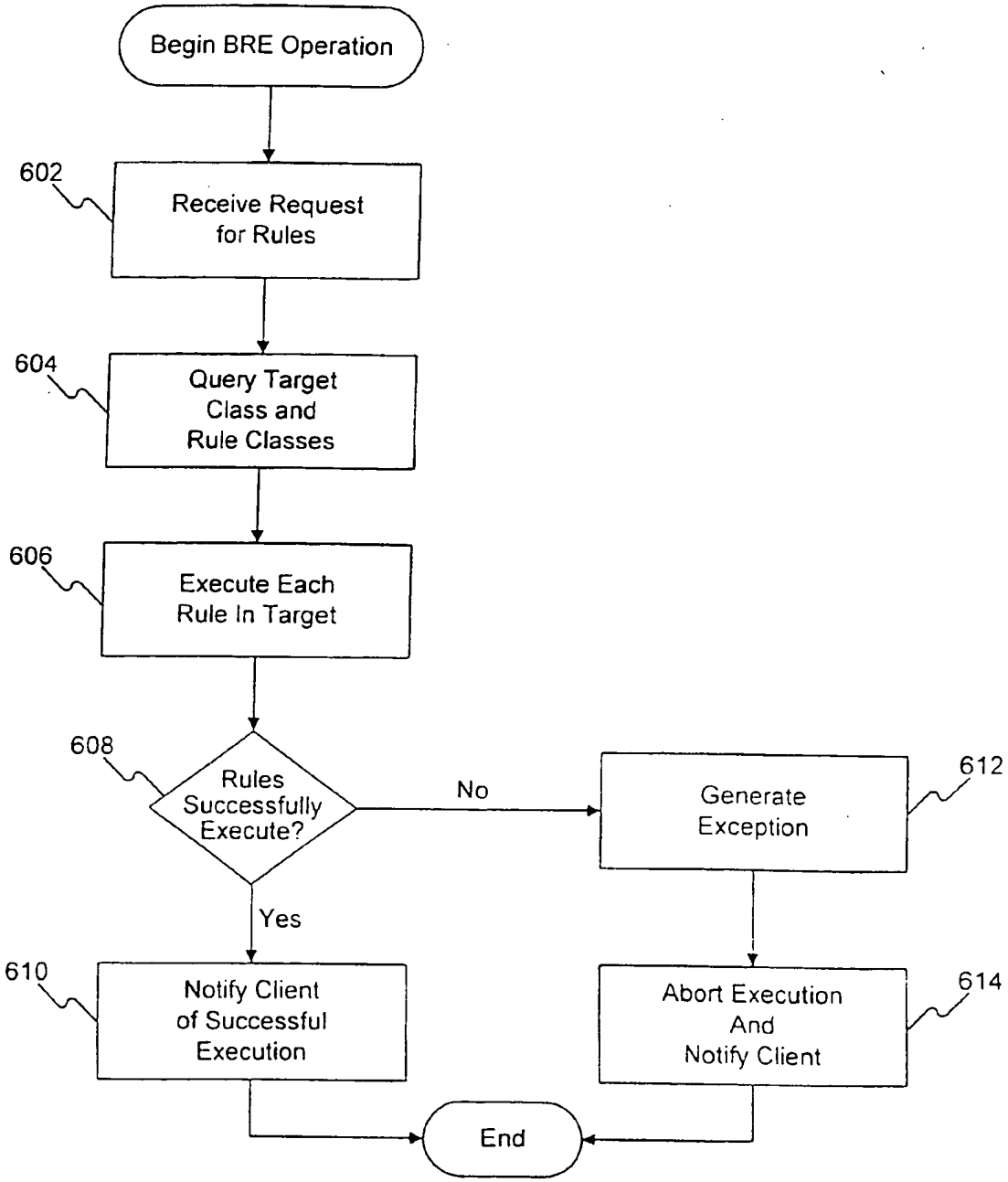

Begin BRE Operation
602
Receive Request for Rules
604
Query Target Class and Rule Classes
606
Execute Each Rule In Target
608
Rules Successfully Execute?
No
612
Generate Exception
Yes
610
Notify Client of Successful Execution
614
Abort Execution And Notify Client
End

FIG. 6